EE769 - Introduction to Machine learning Rendering Natural Camera Bokeh Effect with Deep Learning

Madan Y N
200070040
Electrical Engineering, IIT Bombay
200070040@iitb.ac.in

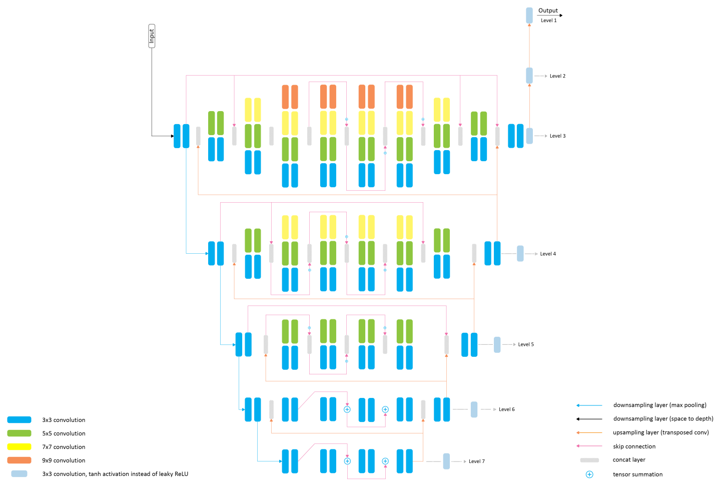
***Abstract*—Bokeh is a significant artistic technique utilized to emphasize the main subject of a photograph by blurring out-of-focus areas. While DSLR and system camera lenses can naturally create this effect, mobile cameras are unable to capture shallow depth-of-field photos due to their optics' very small aperture diameter. In this project, we aim to implement a convolutional neural network (CNN) based deep learning architecture, PyNET, as detailed in the research paper (http://dx.doi.org/10.1109/CVPRW50498.2020.00217), to generate the Bokeh effect. PyNET is designed with an inverted pyramidal shape and processes images at seven different levels. We possess a vast bokeh dataset, including 5,000 shallow/wide depth-of-field image pairs captured using the Canon 7D DSLR with 50mm f/1.8 lenses, which will be utilized to train the deep learning model to produce a natural bokeh effect based on a single narrow aperture image. The model has been trained with up to 400 training images; however, the obtained results are not satisfactory, and the model requires further training. Future work can entail training the model with a larger dataset and fine-tuning the hyperparameters.**

***Index Terms*—Combinatorial optimization, digital annealing, hopfield neural network, memristor, simulated annealing**

I. Introduction

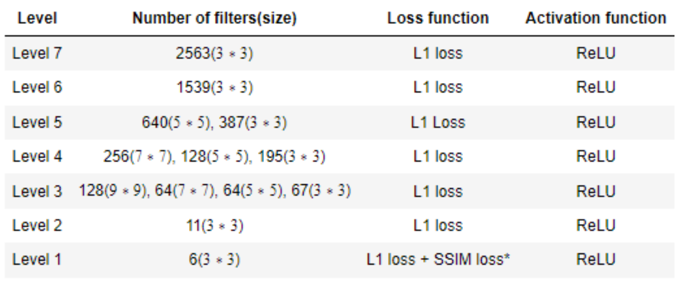
The Bokeh effect is a popular photography technique used to emphasize the subject in the photo by blurring the background. This technique is achieved by using a wide aperture lens and focusing on the selected area or object. However, this effect is challenging to replicate on mobile cameras due to their compact optics and small sensors. Therefore, computational simulation is necessary, and synthetic bokeh effect rendering has emerged as a new machine learning topic. In the paper ”Rendering Natural Camera Bokeh Effect with Deep Learning”, the authors propose an approach to the problem by using deep learning to learn the Bokeh technique directly from photos produced by a high-end DSLR camera. This approach is camera independent, does not require any special hardware, and can be applied to existing images. The proposed method is different from previous approaches, which relied on approximations of the Bokeh effect, and it aims to produce natural Bokeh photos similar to those produced by DSLR cameras.

II. PyNET CNN Architecture



Bokeh effect simulation problem belongs to a group of tasks dealing with both global and local image processing. High-level image analysis is needed here to detect the areas on the photo where the bokeh effect should be applied, whereas low-level processing is used for rendering the actual shallow depth-of-field images and refining the results. A model specifically designed for this purpose is PyNET. It is designed to processing image at different scales and combining the learned global and local features together. The above figure illustrates schematic representation of the PyNET architecture that is used in this project. The model has an inverted pyramidal shape and is processing image at seven different levels. The PyNET-based model architecture includes parallel processing of feature maps with different convolutional filter sizes, concatenation and summation operations on adjacent layers, and the use of transposed convolutional layers for upsampling. The model is trained sequentially from the lowest layer and then applied to the next level until training is done on the original resolution. The model mainly learns to reconstruct missing low-level details and refine the results. The resolution of the produced images is twice higher than the input data size, which increases training and inference speed. Sample visual results show the effectiveness of the proposed deep learning method.

Some specifications of the PyNET model are as follows

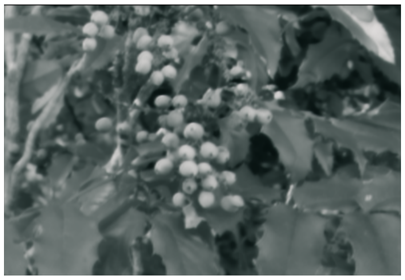


Level	Number of filters(size)	Loss function	Activation function
Level 7	2563(3 * 3)	L1 loss	ReLU
Level 6	1539(3 * 3)	L1 loss	ReLU
Level 5	640(5 * 5), 387(3 * 3)	L1 Loss	ReLU
Level 4	256(7 * 7), 128(5 * 5), 195(3 * 3)	L1 loss	ReLU
Level 3	128(9 * 9), 64(7 * 7), 64(5 * 5), 67(3 * 3)	L1 loss	ReLU
Level 2	11(3 * 3)	L1 loss	ReLU
Level 1	6(3 * 3)	L1 loss + SSIM loss*	ReLU

III. Implementation

The model is implemented using tensorflow keras api and run on google colab gpu runtime. The training details and the results are as follows

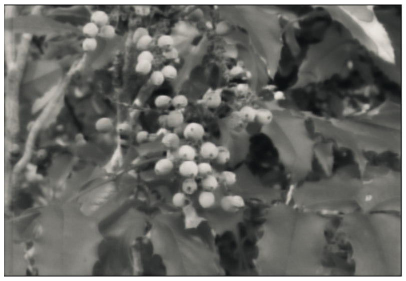
- Training iteration - 1
  The whole model is trained together by optimizing only the level - 1 output. Training dataset had a total of 500 images.
  Result: The performance of the model was unsatisfactory, the model performed poorly on train data itself. An output of the model is given below.



(a) output

(b) target

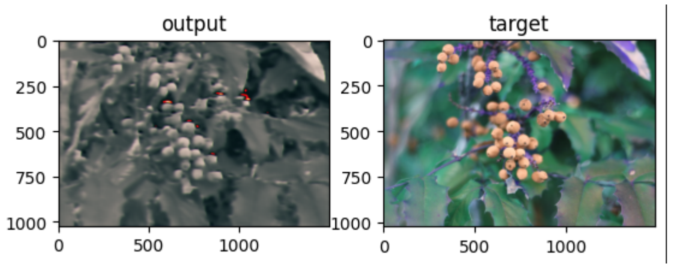
  Proposed solution: Increase the training dataset to 1500 images
- Training iteration - 2
  The whole model is trained together by optimizing only the level - 1 output. Training dataset had a total of 1500 images.
  Result: Although the performance of the model has improved it is still unsatisfactory, the model performed poorly on train data itself. An output of the model is given below.

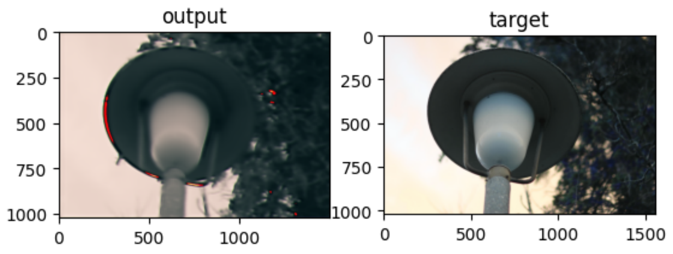


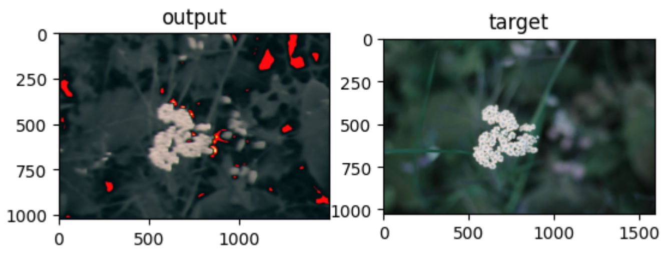
(a) output

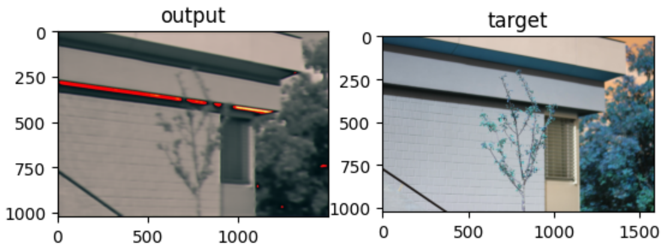
(b) target

  Proposed solution : The paper recommends training the model sequentially by taking all the previous levels as pre-trained. So, train the model sequentially unlike the previous iterations.
- Sequentially training iteration - 1 Model is trained sequentially starting from level-7 all the way till level-1. The training dataset had 100 images. Result: Sequential model with 100 images did not perform well either. Some of the output images are as given below.



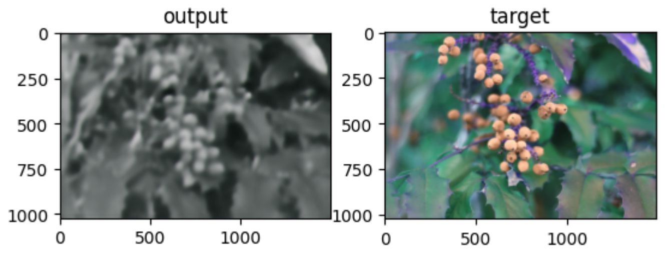


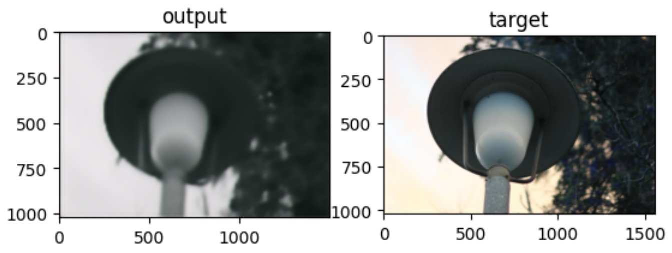


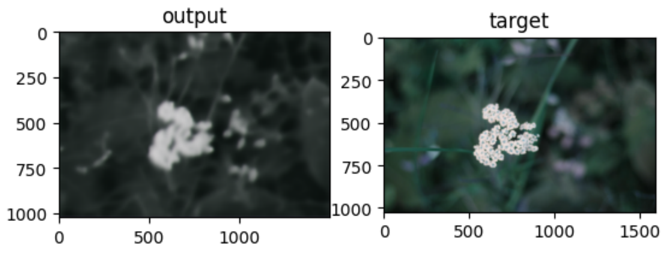


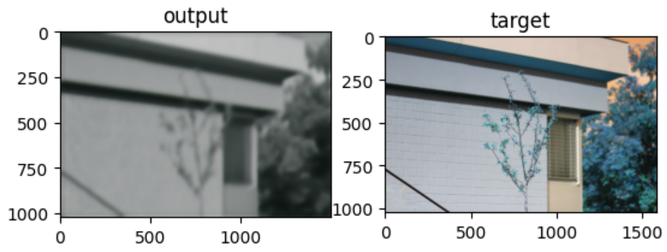
Proposed solution : Possible reasons for the model not performing well can be either the model being too complex or the datset begin too small. We shall try both the solutions

- Reducing model complexity The complexity of the model is reduced by removing the filters of level-3. A total of 323 filters were removed. The treaining dataset had 100 images. Result : The performance of the model further degraded. Some of the output maps are shown below.



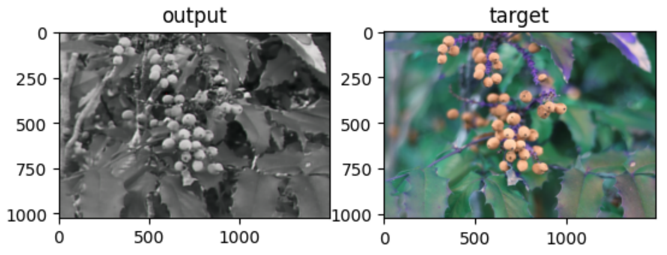


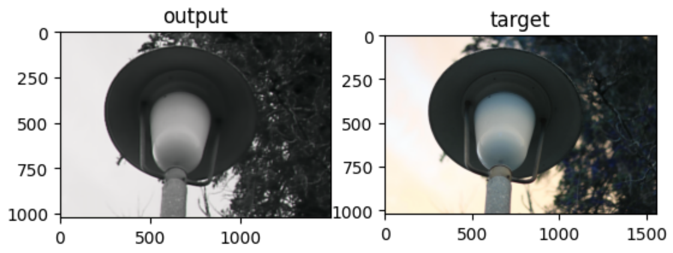


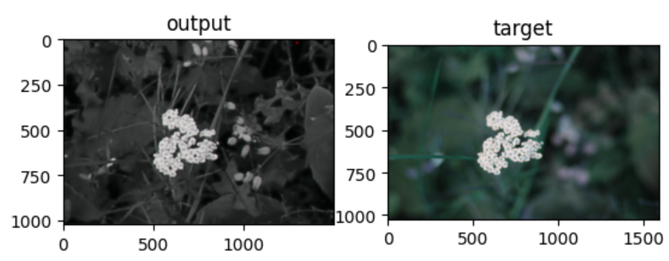


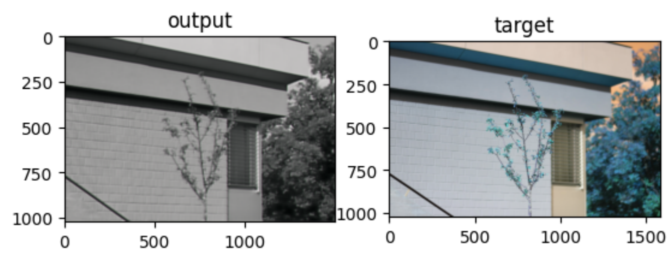
Proposed solution : Reducing model complexity actually degraded the performance of the model. In next iteration of training, training dataset has been increased.

- Increasing training dataset for sequential model
  Model is trained sequentially starting from level-7 all the way till level-1. The training dataset had 400 images. Result: Sequential model with 100 images did not perform well either. Some of the output images are as given below. Result : The four output images shown below show a significant improvement from the model trained with just 200 images. The red artifacts have gone and also the central object is a little more clear than the rest of the image as in the street-lamp image(image 2). Although the output images are still lacking color information, increasing dataset seems promising









IV. Conclusions and Future Work

The project is partially complete. I did not have access to any gpu locally I used google colab and since there is a limit on use time in google colab I could not get enough computational resources. The model should be trained with a larger dataset and this will enhance the result. Future work will start from training the model with a larger dataset, tuning the hyperparameters and finally deploy the model as a web application.

V. References

1. https://github.com/aiff22/PyNET-Bokeh - Master project repository of the research paper which I have implemented. Motivation of the code and high level idea of the PyNet function is taken from this source
2. https://www.tensorflow.org/api-docs - Official tensorflow api documentation. I reffered to to this resource while using the tensorflow library functions like Conv2D, ConvTranspose2D, etc. 3. https://chat.openai.com - ChatGPT. I used chatgpt to clear my doubts about understanding concepts that could not done from the above resources. I used it to understand how tf.sessions work, etc

Github - https://github.com/MadanYN/EE769-Project.git
Recording - https://drive.google.com/folderview?id=1qGxiX8-40r9kpMg21YTmto21KBYA07zN